JAVA EMBEDDED SERVER SOFTWARE

White Papers

Java Embedded Server

- White Papers
- » Documentation
- » FAQs

Related:

- » In the Spotlight
- » In the News
- » Testimonials
- » Technical Support
- » How to Buy
- » Connected Home

Anne Thomas
Patricia Seybold Group
www.psgroup.com

October 1998
Prepared for Sun Microsystems, Inc.

---

1. Executive Summary
Java Embedded Server

Sun Microsystems' Java Embedded Server (JES) provides a powerful and versatile utility to support the embedded application environment. Embedded applications run on microcomputers that are embedded within electronic devices such as appliances, vending machines, gasoline pumps, cellular phones, or pagers.

Extending Corporate Applications to the Devices

Until recently, these embedded computers were extremely constrained in terms of space, power, memory, and network connectivity. But in accordance with Moore's law, these constraints have relaxed, and embedded systems have become much more powerful. Organizations are beginning to capitalize on the increased capabilities of embedded microprocessors by building powerful new embedded applications that extend the reach of corporate application systems directly to the device. For example, a printer can communicate with its network management system to proactively order more paper or toner when levels reach a low threshold. A smart cellular phone or interactive pager can interoperate with a host-based sales force automation system. Or a vending machine can interoperate directly with a distribution system to order replacement supplies and to record consumption market data. The embedded software market is on the brink of massive expansion.

On-Demand Java Applications

Java is one of the most popular programming languages for building these new embedded applications. JES is an elegant, small-footprint application server especially designed for the embedded software market. JES lends a number of powerful benefits to this market. Using JES, a device can dynamically download a Java application, install it, and execute it on demand. When the application is complete, JES purges the application to make room for the next request. Although microprocessors are bigger than ever before, there are still memory constraints that limit the number of applications that can be concurrently installed on the device. JES enables a device to dynamically install any number of applications, dramatically increasing the versatility of the device. JES also extends the useful life of a

device. In the past, a feature upgrade often required a new device. But, with JES, new features can be added through software. JES automatically upgrades the device software the next time the application is loaded.

Back to Top

---

2. Introduction: Device-Based Embedded Applications

Electronic Devices and Embedded Microcomputers

When you say the word "computer," most people think of desktop systems, server systems or mainframes. But computers also run within a wide range of electronic devices, such as network routers, printers, robots, pagers, telephones, vending machines, home appliances, locks, and automobiles. The embedded microcomputers control the operation of these devices.

Real-Time Operating Systems

Many of these microcomputers use a specialized operating system that can respond to events in real time, that is, in milliseconds rather than seconds. Most general purpose operating systems, such as Unix, rely on a clock that measures time in seconds. A real-time operating system (RTOS) uses a clock that measures time in milliseconds, and it can respond to events much more quickly than Unix can. For example, an electronic ABS braking system in an automobile must respond as soon as the driver presses the brake pedal-it can't wait for the next click of the second hand.

Specific and Limited Functions

In addition to reaction time, space is a primary concern within embedded systems. In recent years, size and capacity have severely limited the capabilities of these microprocessors. Most embedded systems have used 8-bit or 16-bit processors with minimal memory. The devices were often programmed using hardware-specific microcode, and they could perform only a limited number of functions. Upgrading the embedded applications often required upgrading the entire device.

Moore's Law in Action

But the cost and size of microprocessors and memory boards has been steadily falling in accordance with Moore's Law, and the former capacity considerations are much less critical now than they were just a couple of years ago. Significantly more processing power and memory can fit into the same space, and these embedded processors are becoming much more intelligent and versatile. Network devices that once could perform only one or two specific functions can now be programmed to perform a variety of operations. Vendors are capitalizing on these capabilities and generating a new industry segment known as *thin servers*.

Thin Servers

A thin server is a smart electronic device that can execute operations in response to client requests. For example, the latest generation of networking devices (routers, bridges, controllers, etc.) generally includes a microWeb server that enables remote administration. An administrator working from any Web browser can monitor a device, query statistics, and change settings through a standard HTTP connection. Printers, copiers, and fax machines now offer a variety of application and control options accessible through the network.

New Application-Specific Devices

Embedded systems are also being used to develop a new line of sophisticated application-

specific devices such as personal data assistants (PDAs), smart pagers, and Web phones. The latest Web phones now support a variety of useful applications, such as an address book, an auto-dialer, an Internet browser, and a calendar.

Embedded Java

Java is one of the more popular tools for building applications for embedded systems. Java, in fact, was originally created with embedded application systems in mind. A small-footprint Java virtual machine, called Embedded Java, can be installed on a microprocessor. Rather than using complex programming systems such as microcode, Assembler, or C, developers can build these control applications in Java.

Java Benefits

Java lends a number of very valuable benefits to the embedded application space. Java is a simple and productive programming language that can reduce the time and effort required to develop embedded applications. Like other Java environments, embedded Java applications can be ported and reused in other devices. And perhaps the most intriguing benefit is that Java applications can be downloaded across the network, installed, and executed on demand.

On-Demand Applications

Although memory is not quite as scarce as it once was, embedded systems still have limited local memory resources. Only so much space is available for pre-installed applications. But if applications can be loaded on demand, then a small microprocessor can become a much more versatile computing system. Where once a device could perform only one or two operations, now it can perform a wide variety of operations.

Simple, Flexible Management

This approach to embedded applications simplifies management of the devices. The applications can be maintained and administered in a centralized location. And users are no longer required to replace the entire device in order to upgrade to new services or capabilities. They simply load a new version of the controlling Java application.

Back to Top

---

3. Java Embedded Server Application Server

Sun Microsystems' Java Embedded Server is small-footprint application server designed specifically to support on-demand embedded applications. The Java Embedded Server allows a device to dynamically download and execute an application from a remote location on the network.

System Architecture

Java Embedded Server consists of two primary components: the ServiceSpace and the Services. The ServiceSpace, only 100KB in size, provides a runtime framework that manages the loading, installation, activation, execution, and removal of applications, called *services*. The Services are a set of useful pre-built and customizable services. Device providers and developers can build their own services using the Java Service Designer tool.

The illustration shows the architecture of the environment. The ServiceSpace is installed

on a device running an embedded operating system. ServiceSpace requires a Java runtime platform, such as EmbeddedJava, PersonalJava, or the standard JDK. The services can be stored anywhere on the network, and they are downloaded and executed as required.

Back to Top

Java Embedded Server

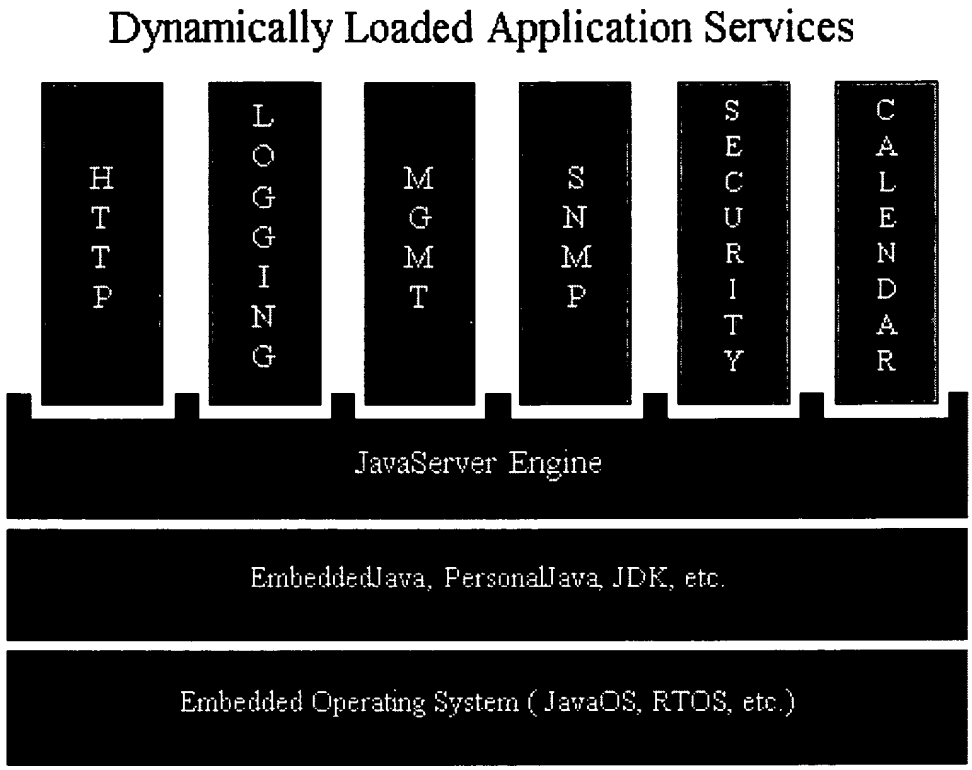

Illustration. Java Embedded Server is a small-footprint dynamic application server designed especially for the embedded application market. Java Embedded Server consists of two primary components. The ServiceSpace runs within a Java runtime platform (EmbeddedJava or PersonalJava) on a network-enabled device. The engine manages the loading and execution of application services. The Services include a set of pre-packaged application services, including a lightweight HTTP server, an SNMP agent, a scheduling service, and a Servlet manager.

Services

Applications that run in the Java Embedded Server are called *application services*. The Java Embedded Server comes with a set of pre-built services that address a variety of common requirements. The Services include:

- **HTTP.** The lightweight HTTP service provides a complete embedded Web server. Developers can use the HTTP service to build a Web interface to the device.
- **Servlets.** The Servlet manager supports the execution of Java Servlets, which can be used to dynamically generate HTML for the embedded Web server.
- **Remote Management.** The remote management service supports remote management of application services resident on the device. The remote management service can be integrated with enterprise management systems, such as IBM Tivoli and CA Unicenter.
- **SNMP.** The SNMP agent service can publish standard or custom SNMP MIBs to enable integration with popular SNMP management frameworks.

- **Scheduler.** The time-based scheduling service can be used to automatically activate an application service to run at a particular time.
- **Logging.** The logging service can track events and errors and write a simple text log.
- **RMI.** The remote method invocation service allows an application service to invoke a method on a remote object using standard RMI.
- **Console.** The console service provides an administrative console for the Java Embedded Server.

Example Application Services

Any potential embedded application could be implemented as a Java Embedded Server application service. Installing the Java Embedded Server on the device allows any application service to be loaded dynamically. Here are a few example applications:

- A printer security application, which prohibits the printing of confidential documents, could be loaded on all printers located in unsecured areas.
- A replenishment application could be loaded onto gasoline pumps or vending machines. When inventory within the device reaches minimum threshold levels, the application service automatically calls a host-based distribution system to schedule delivery of more stock.
- A device configuration application could be loaded into robotic manufacturing equipment. System administrators could easily reconfigure the device control systems from a browser. This type of application would be especially valuable to any system that requires clean-room operations, since humans hands would not need to touch the device for reconfiguration.
- A software distribution and versioning control system could be loaded into any device to ensure that the device is always using the latest version of an application service. Each time an application service is launched in the device, the software distribution system would check to see if a new version is available. If the software has been upgraded, the software distribution system would automatically download and install the new upgrade.

Back to Top

---

4. ServiceSpace
Service Deployment Framework

The ServiceSpace provides the heart of the Java Embedded Server. The ServiceSpace provides a service deployment framework that manages the lifecycle of the application services. The framework makes it easy to develop and deploy on-demand embedded applications, and it provides a set of standard programming interfaces that enable portability and consistency across multiple devices.

Lifecycle Services

The ServiceSpace includes a lightweight registry that can dynamically map a Java Interface name to an application implementation. ServiceSpace can locate the application, download it to the device, install it, customize it, and dynamically establish connections between it and other related applications. When the application is no longer needed, the ServiceSpace automatically removes the application from the device. Java Embedded Server supports automatic thread and connection management.

Service Discovery and Dependency Resolution

Application services often have dependencies on other application services. For example, when an error management service identifies a problem, it depends on a paging service to send a page to a service technician. Java Embedded Server provides a dynamic service

discovery mechanism. When an application service is loaded, it identifies all of its service dependencies, and the ServiceSpace determines which services are available and active. If a desired service is not available, the ServiceSpace can locate and load the requested service.

Required and Optional Dependencies

A service dependency can be either required or optional. If the dependency is required, then the service will not be able to perform properly without the associated service. If the dependency is optional, then the service can perform adequately, but the associated service will enhance its capabilities. Java Embedded Server can automatically adapt to take advantage of services when they are available.

Web Phone Example

A Web phone provides a good example of this capability. A Web phone can support a wide variety of personal productivity applications, such as a Web browser, an auto-dialer, an address book, a telephone directory, a yellow pages directory, a calendar, an activity log service, a notebook, a to-do list, and a billing service. These applications can be loaded individually, or they can all be loaded at once. Each of the applications can work effectively on its own, but when integrated, these applications become much more powerful. The auto-dialer can dial any number specified by the user. It can also be enabled to automatically make use of the other available application services. The auto-dialer can retrieve a phone number from the address book, the telephone directory, the yellow pages, or a Web page. When it makes the call, it can invoke the activity log service to record the call, noting its date, time, and duration; the identity of the person placing the call; the phone number; and the person being called (if known). The activity log service can automatically place an entry in the calendar. The calendar can pop up the notebook to allow the user to record personal notes to attach to the call record. The calendar might also prompt the user to specify any to-do activities, which, in turn, get recorded in the calendar. When the call is complete, the billing service calculates the accrued charges.

Application Service Bundles

An application service is deployed using an application service *bundle*. A bundle comprises a Java archive (JAR) file and a set of instructions for the Java Embedded Server. The instructions come in the form of Java objects called *wizards*. Each bundle contains the following:

- **Manifest.** The manifest defines the contents of the bundle.
- **Installer Wizard.** The installer wizard installs the service on the server.
- **Activator Wizard.** The activator wizard starts the service and registers it with the server.
- **Update Wizard.** The update wizard automatically manages version control and replaces older versions of the service with the latest one.
- **Dependencies Wizard.** The dependencies wizard identifies any dependencies this service may have on other services and determines if the services are available. The dependencies wizard can optionally install other required services that are not yet installed on the server.
- **Content.** The content contains Java classes and other files used by the service.

Back to Top

---

5. Conclusions

New Class of Application Server

The Java Embedded Server defines a new class of application server for the potentially explosive new market of thin servers and Internet-enabled devices. A small number of

vendors provide embedded Web servers for this market, but no other vendor provides an application server that supports the dynamic loading of embedded application services on demand.

Small Footprint

Java Embedded Server requires 100KB for the basic system, plus additional memory for each concurrently loaded application service. Java Embedded Server also requires a Java platform, such as EmbeddedJava or PersonalJava. A typical runtime environment would require less that 1MB of memory to support the operating system, the application server, and the application services. At the moment, these memory requirements still seem fairly heavy for many devices, although memory limitations are becoming less of an issue in modern embedded systems as miniaturization techniques continue to improve. Within a few years, Java Embedded Server will fit within all but the tiniest embedded systems.

Increased Device Versatility

Java Embedded Server increases the versatility of Internet-enabled devices. Modern microprocessors are powerful computer systems that can be programmed to do a variety of applications and services. But space and memory limitations restrict the number and size of applications that can be loaded into the device. By loading and removing applications on demand, a single device can be extended to support any number of different operations.

Extending Device Usefulness

Java Embedded Server extends the usefulness of intelligent devices by enabling simple and easy upgrades to the software that controls the device. In the past, the device software was tightly integrated with the hardware, and an upgrade generally required the purchase of a new device. Java Embedded Server provides inherent support for version control and automatic updates.

Out-of-the-Box Services

Java Embedded Server comes with a number of useful pre-packaged application services, such as an embedded Web server, management interfaces, and an administration console. The basic platform provides a feature-rich environment to help device providers and developers get started quickly.

Enabling a New Consumer-Based Device Industry

Java Embedded Server is likely to have a major impact on the new emerging consumer-based device industry. Very few home appliances currently support interactive interfaces and network capability. But new appliances are being developed to take advantage of Internet connections in the home. Initially, we will see Web phones and Internet-enabled televisions. Over time, many other appliances and devices will join the network.

Enabling Jini-Based Federations

Jini, a federated Java integration service, has gained a lot of attention recently. Jini provides a loosely coupled mechanism for applications on different devices to communicate and share services. But Jini doesn't provide a mechanism to dynamically load application services within a device. The two technologies are complementary. Java Embedded Server could provide a mechanism to dynamically load Jini services. The possibilities are very exciting.

In business since 1978, the Patricia Seybold Group provides strategic guidance and tactical advice for organizations seeking business advantage through the application of information technology. The Group has built an international reputation for excellence and objectivity in its research and analysis, and for the provision of consulting services which identify strategies and tools best suited to the development of the client's unique business and technology needs. The company office is located at 85 Devonshire St, 5th floor, Boston, MA 02109. For further information about our publications and research services, please visit our web site (www.psgroup.com). For information about consulting services, please contact Ed Detrick or Deb Hay at 617.742.5200, email: edetrick@psgroup.com or dhay@psgroup.com.